A MULTIMODAL CONVERSATIONAL AI SYSTEM FOR INTELLIGENT REAL-TIME ANDROID INTERACTION

Ms.Chitra K
Assistant Professor
Department of Computer Science and Engineering
Bharath Institute of Higher Education and Research
Chennai, India

Deepika R
Department of Computer Science and Engineering
Bharath Institute of Higher Education and Research
Chennai, India
rengarajdeepika189@gmail.com

Dinesh V
Department of Computer Science and Engineering
Bharath Institute of Higher Education and Research
Chennai, India
dineshdeen1812@gmail.com

Eswar S
Department of Computer Science and Engineering
Bharath Institute of Higher Education and Research
Chennai, India
skeswar474@gmail.com

Farheena K
Department of Computer Science and Engineering
Bharath Institute of Higher Education and Research
Chennai, India
farheenafarheena7@gmail.com

Abstract*— Artificial Intelligence (AI) has significantly transformed Human-Computer Interaction (HCI) by enabling natural language-based communication between humans and machines. Voice assistants are helpful tools that make our daily tasks easier and less complicated when we use digital systems. However, most existing assistants suffer from limited personalization, lack of contextual understanding, platform dependency, and poor real-time conversational flow. This project introduces a conversational AI voice agent designed to understand and respond to users through speech in a real-time manner across both desktop and Android mobile platforms.

Keywords*—HCI, Large language model (LLM), API, TTS, Conversational AI, LiveKit, Gemini, Voice Assistant, Speech Interaction, NLP, Android, Realtime Streaming.

I. INTRODUCTION

In recent years, human–computer interaction has undergone a massive transformation. Traditional input methods such as keyboards, mice and touchscreens have gradually been replaced by intelligent voice-based interfaces.

Conversational AI is a technology that helps computers understand what we are saying. The goal of this project is to create an AI system that is easy to use and can understand what we want. This conversational AI system can work on both desktop and Android mobile platforms.

Unlike conventional voice assistants such as Siri or Alexa, the proposed system is designed with open-source flexibility. It can be fully customized, run locally for maximum privacy and integrated with different modules to perform a wide range of functions. It utilizes Natural Language Processing (NLP) to interpret user input and Machine Learning (ML) to improve response accuracy over time.

II. LITERATURE SURVEY

In this literature survey, we have looked at different voice assistant systems and how they work. We have also looked at their cons. Based on what we have learned we are trying to create an AI system that is better at understanding what we want and responding to us in real time.

[1] In this paper, the authors present a voice assistant system developed using Artificial Intelligence techniques that can interpret user speech and respond using synthesized voice. The system follows a basic speech recognition, keyword extraction and execution of predefined commands such as sending messages, copying files and performing web-based tasks. While it improves user convenience through voice interaction, the system is limited by its static nature and reliance on predefined commands without advanced contextual understanding.

[2] This study explores the development of voice assistants integrated with machine learning, Natural Language Processing (NLP) and IoT technologies. The system demonstrates how assistants can process large-scale data and perform smart operations such as home automation and device control. However, the implementation complexity and dependency on multiple technologies make the system less efficient for real-time applications.

[3] In this research, an Android-based voice assistant application is developed using keyword matching and basic NLP techniques. The system uses Android Speech Recognizer for voice input and Text-to-Speech (TTS) for output, enabling functions like calling, messaging, alarms and app control. Although it ensures better privacy, its intelligence is limited due to lack of deep conversational capabilities.

[4] This paper discusses the evolution of voice assistants like ELIZA to modern cloud-based assistants such as Siri and Google Assistant. It highlights improvements in speech recognition and NLP, but it also talks about major limitations such as their need for internet connectivity and their lack of customization.

[5] Finally we looked at a paper about how to make voice assistants better. It talked about using a combination of online processing to make voice assistants more efficient and effective. It also talked about the challenges of creating voice assistants that can understand what we want in time and respond to us in a natural way.

III. RELATED WORKS

Voice assistants have emerged as an essential component of modern Human–Computer Interaction, enabling users to control applications and smart devices through speech. They are convenient, accessible and support multitasking environments. Major AI-powered assistants such as Siri, Google Assistant and Alexa uses speech recognition and natural language processing technologies, designed for consumer electronics. These systems represent a big industry. However, despite their popularity, they have limitations still exist in personalization, conversational liberty and execution of real-time task automation, indicating a need for more flexible research-driven systems that upgrade scalability, emotional interaction and open development.

1) Cloud-Based Conversational Assistants

Siri and Google Assistant are known commercial voice assistants that rely heavily on cloud-based Natural Language Processing for command interpretation and service execution. While these systems deliver high-quality speech recognition, they lack personality customization and restrict developers from integrating advanced external tools. Most responses remain pre-decided and lack tone, resulting in robotic and dull interaction. Additionally, privacy challenges and network dependency contribute to performance variations under connectivity conditions.

2) Continuous Dialogue and Context Handling

Early conversational systems were primarily rule-based chatbots, which struggled to retain context across long conversations. The traditional natural language processing workflow requires predefined patterns and often fails when presented with input or humor-based dialogue. This leads to breakdowns in communication and user dissatisfaction. Advanced Large Language Models-driven dialog systems have improved retention, yet many implementations remain text-focused without real-time speech support.

3) Real-Time Streaming Architectures

Recent works with frameworks like LiveKit showing improvements in streaming latency and communication. Listening environments require optimized packet delivery and silence detection to maintain a natural flow of dialogue. However, this field is still developing and only a few academic projects have combined real-time streaming with personality features. JARVIS stands out from work by combining real-time speech, personality-driven responses and external tool execution in a modular design. This approach bridges the gap between what commercial systems can do and what academic models offer providing an advanced solution that enhances natural user engagement while maintaining useful automation capabilities.

IV. PROPOSED SYSTEM

Our system introduces an intelligent conversational voice assistant designed to enable smooth communication between humans and computers through real-time voice interaction. It removes the dependency on input devices by continuously listening, processing, and responding to user speech. The architecture is modular, scalable and supports enhancements such as IoT integration and advanced personalization features.

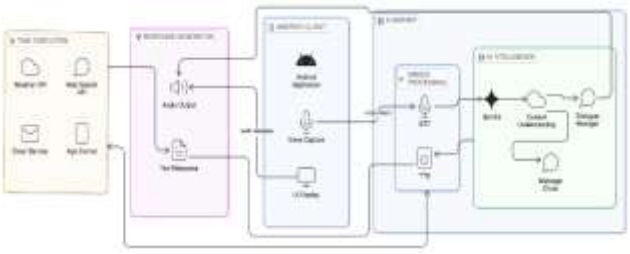

Fig.1. System architecture

In this project, a real-time conversational assistant is developed along with an Android application to provide smooth voice-based interaction. The Android application is built using the Livekit Android SDK in Android Studio, where a custom voice assistant interface is created to capture user speech through the mobile device. The app handles microphone input, permissions and network

communication and connects directly to the Live Kit server to stream audio in real time. This ensures low-latency, bidirectional communication between the user and the backend system.

The captured voice input is transmitted through Live kit to the backend, where the Gemini language model processes it. The system performs speech-to-text conversion, analyzes the user's intent using Natural Language Processing (NLP), and generates an appropriate response. Based on the request, the system can execute tasks such as retrieving weather information, performing web searches or triggering automation functions. The generated response is then converted back into speech and streamed to the Android application, allowing the assistant to respond naturally to the user.

A key feature implemented in the system is a personalized conversational style, where the assistant responds with a distinct tone, making the interaction more engaging and human-like. The modular design allows easy integration of new features such as reminders or context-aware responses in future updates. By combining Android-based mobile interaction, real-time streaming through LiveKit, and intelligent processing using advanced language models, the system provides an efficient and user-friendly approach to voice-driven Human–Computer Interaction (HCI).

The system works like a helper that can understand what people are saying, keep talking to them, do things in time and answer quickly. It fills the gap between voice assistants and new ones that are more like people. The system is flexible. Can be used in the future. The conversational assistant and the Android application work together to make an experience for the user. The LiveKit server and the Gemini language model are parts of the system. They help the conversational assistant understand. Respond to the user.

A. System Pipeline

The voice assistant has a system pipeline that shows the flow of data from user input to system response in the voice assistant. It starts with the Android application capturing the user's speech through the microphone and streaming it to the backend using LiveKit for real-time communication. The audio is then processed in the AI layer, where it is converted into text and analyzed using the language model to understand the user's intent.

Fig.2. System pipeline

Based on the user request, the system pipeline may trigger the right tools such as weather services or web search to fetch information. The language model then generates a suitable response, which is converted back into speech using text-to-speech. The voice assistant uses the system pipeline to get the answer, to the Android application and the user gets to hear it, which completes the whole process of talking to the voice assistant and getting an answer. The system pipeline is a part of how the voice assistant works.

B. Project Dependencies

1. MainActivity (Java / Kotlin)

PURPOSE: Controls the app and handles user interaction.

Fig.3. MainActivity.kt

2.. XML Layout Files

PURPOSE: Defines the user interface design.

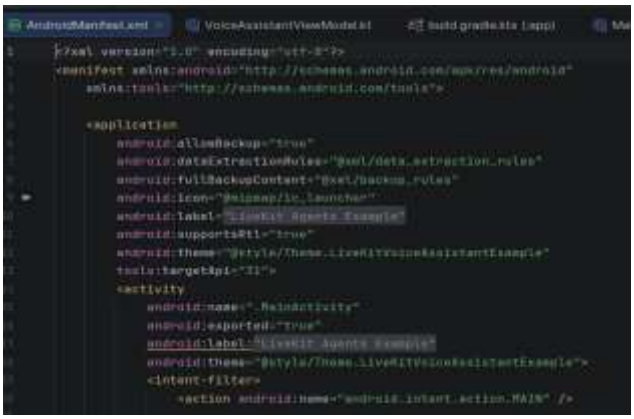

Fig.4. AndroidManifest.XML

3. Network Module

PURPOSE: Connects app with backend system.

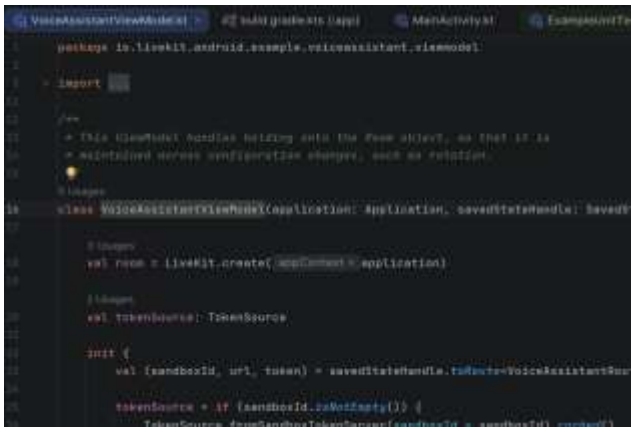

Fig.5. VoiceAsssitantViewModel.kt

4. Permissions (Mic, Internet)

PURPOSE: Allows access to required device features.

Fig.6. Permissions.kt

C. User Interaction Output Demonstration.

1. APPLICATION LAUNCH ROOM

The Android voice agent application launches with a splash screen displaying the LiveKit voice assistant icon. This means the app was installed and set up correctly on your device. The interface has buttons for Talking to the agent using voice input, accessing the camera, chatting with text, sending messages and ending the assistant session.

Fig.7. Real-time Android App

2. AGENT CONNECTED

Voice Activity Visualize Upon successful connection to the LiveKit room, the agent displays an animated waveform visualizer indicating active voice streaming between the user's smartphone and the AI agent in real time. The bars move up and down when the agent is speaking or listening. This helps you see when the agent is active.

3. CHAT INTERFACE

Below are some examples of how our system can understand and respond to user queries. When the user asks about trending chatbots, the assistant processes the request and provides a meaningful and contextual response, highlighting its natural language understanding capability.

Fig.8. Information Retrieval

4. CONVERSATIONAL INTERACTION

The system engages in natural conversation with the user. It understands informal inputs such as project-related discussions and responds intelligently.

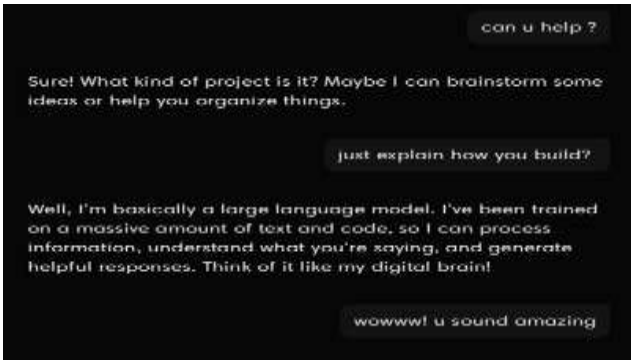

Fig.9. Agent Interaction

5. WEATHER RETRIEVEL

Our agent successfully retrieved and responded with the weather information for Mumbai, India. It also demonstrated contextual memory by recognizing that the question had been asked previously.

Fig.10. Weather output

This confirms that our system can accurately get weather information and have a conversation.

A. Backend Implementation

The terminal output of a LiveKit agent during execution. It illustrates the initialization of the job runner, followed by system-generated logs indicating that the worker has reached its load capacity. As a result, the system marks the worker as unavailable, initiates a draining process and subsequently shuts down the worker. This output demonstrates how the system manages resource limitations and ensures stability by preventing overload during real-time processing.

system execution logs — python agent.py

Fig.11. Back-end implementation

B. Frontend Implementation

The front-end smoothly handled user input and displayed expressive AI responses in real time.

Fig.12. Front-end implementation

The pictures show the front-end of the Android application developed for the project, designed to support real-time conversational interaction. The app integrates with LiveKit to enable voice-based communication, where animated visual indicators represent active audio processing. It also includes a chat interface for text-based conversation, displaying user inputs and system-generated responses. The application provides essential features such as microphone control, messaging and call management, ensuring a smooth, interactive and user-friendly experience for both voice and text communication.

V. RESULT AND DISCUSSION

Our voice assistant system demonstrates efficient performance in real-time conversational scenarios, successfully integrating speech input, intelligent processing and natural voice output into a unified pipeline. The results indicate that the system is capable of accurately recognizing user speech, interpreting intent and generating meaningful responses with minimal delay. The use of a real-time AI model enhances the conversational quality, allowing the assistant to respond in fluent and human like manner while maintaining consistent personality-driven interaction style.

From a performance perspective, the system shows low latency and stable communication due to the integration of LiveKit for real-time audio streaming. The assistant can maintain continuous interaction without noticeable lag, even during longer conversations.The modular architecture improves system flexibility, enabling easy addition of new features such as external tools and automation services. This scalability makes the system suitable for future enhancements and broader applications.

In terms of usability the assistant provides an experience through voice interaction eliminating manual input. Testing results confirm performance with good speech recognition accuracy and consistent responses. Overall, our system successfully combines AI, real-time communication and modular design to create an effective voice assistant.

VI. FUTURE SCOPE

EMOTION AND SENTIMENT RECOGNIZITON
Future versions of our system can analyze vocal tone and sentiment to respond empathetically, making interactions more natural and emotionally intelligent.

MULTILINGUAL SUPPORT
Integrating translation and multilingual NLP models will enable the assistant to communicate with users in different languages.

CLOUD - EDGE HYBRID MODEL

A hybrid model could balance performance and privacy by processing sensitive tasks locally while using cloud resources for complex computations

ADVANCED PERSONALIZATION
The system can use user profiling to understand preferences and provide personalized responses. This improves user experience by adapting to individual behaviour over time.

EMOTION & SENTIMENT DETECTION
Sentiment analysis can be added to detect user emotions from speech or text. This helps the assistant respond in a more natural and empathetic manner.

OFFLINE FUNCTIONALITY
Basic features can be made available offline without internet dependency. This ensures usability in low or no connectivity conditions.

IMPROVED AI MODELS
Upgrading to advanced AI models will enhance accuracy and contextual understanding. This results in more intelligent and human-like conversations.

REAL-TIME DATA EXPANSION
Integration of more APIs can provide real-time data like news and navigation. This makes the assistant more informative and practical for daily use.

OFFLINE MODE WITH LOCAL LLM DEPLOYMENT
Future versions can include lightweight on-device language models to enable essential functions even without an internet connection, improving accessibility and reliability.

VII. CONCLUSION

The Voice Agent project successfully demonstrates how modern conversational AI can really change the way people interact with computers using natural voice communication. This project was built using free and open-source tools like Python LiveKit Agents SDK and Google Gemini 2.5 Flash Native Audio. The Voice Agent system can understand what people say do things with seven tools like checking the weather, searching the web, sending emails and using the camera. It can even talk back to people in a way and change its tone based on how they behave. The Voice Agent project also works on Android mobile devices using LiveKit WebRTC, which means that people can use it on their computers and phones without spending any money. The Voice Agent is an example of how a smart AI voice assistant can be built for free and work on many different platforms. This project can be improved in the future to work offline connect with devices and use voice recognition to identify people. The Voice Agent project is a starting point, for making even better AI voice assistants in the future.

VIII. REFERENCES

[1] S. V. Doshi, S. B. Pawar, A. G. Shelar, and S. S. Kulkarni, "Artificial Intelligence Chatbot in Android System using Open-Source Program-O," International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), vol. 6, no. 4, Apr. 2017.

[2] P. G., et al., "Voice Assistant using Artificial Intelligence," International Journal of Engineering Research & Technology (IJERT), vol. 11, no. 05, May 2022.

[3] K. Yasodha, et al., "AI Voice Assistant Android App using Keyword Matching and NLP," International Journal of Computer Techniques (IJCT), vol. 12, no. 4, Jul.–Aug. 2025.

[4] F. J. Kiwa, et al., "AI Voice Assistant for Smartphones with NLP Techniques," in IEEE 2nd Zimbabwe Conference on Information and Communication Technology (ZCICT), 2023.

[5] C. González-Mora, et al., "Augmenting Websites with Voice Commands: An Approach Focused on Accessibility," Journal of Web Engineering, vol. 24, 2025.

[6] "Desktop Assistant AI Using Python," International Research Journal of Engineering and Technology (IRJET), 2020.

[7] P. Shende, et al., "AI Based Voice Assistant Using Python," International Journal of Computer Applications, 2020.

[8] R. Geetha, et al., "The Voice Enabled Personal Assistant for PC Using Python," International Journal of Engineering Research & Technology (IJERT), 2021.

[9] S. Kulhalli, et al., "Personal Assistant with Voice Recognition Intelligence (PARI)," International Journal of Innovative Research in Computer and Communication Engineering, 2020.

[10] "Desktop Assistant AI Using Python," International Research Journal of Engineering and Technology (IRJET), 2020.

[11] Shende, P., et al., "AI Based Voice Assistant Using Python," International Journal of Computer Applications, 2020.

[12] Geetha, R., et al., "The Voice Enabled Personal Assistant for PC Using Python," International Journal of Engineering Research & Technology (IJERT), 2021.

[13] Kulhalli, S., et al., "Personal Assistant with Voice Recognition Intelligence (PARI)," International Journal of Innovative Research in Computer and Communication Engineering, 2020.

[14] Terzopoulos, G., and Satratzemi, M., "Voice Assistants and Artificial Intelligence in Education," International Journal of Emerging Technologies in Learning (IJET), 2019.

[15] Malodia, S., Islam, N., et al., "Why Do People Use AI-Enabled Voice Assistants?" Journal of Retailing and Consumer Services, 2021.